Automatically mimicking unique hand-drawn pencil lines

AlMeraj, Zainab; Wyvill, Brian; Isenberg, Tobias; Gooch, Amy A.; Guy, Richard

Published in:
Computers & Graphics

DOI:
10.1016/j.cag.2009.04.004

Publication date:
2009

Computers & Graphics 33 (2009) 496–508

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Computational Aesthetics 2008

Automatically mimicking unique hand-drawn pencil lines

Zainab AlMeraj [a,b,e,*], Brian Wyvill [b], Tobias Isenberg [c], Amy A. Gooch [b], Richard Guy [d]

[a] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
[b] University of Victoria, Canada
[c] University of Groningen, The Netherlands
[d] University of Calgary, Canada
[e] Kuwait University, Kuwait

ARTICLE INFO

Article history:
Received 2 December 2008
Received in revised form
15 April 2009
Accepted 16 April 2009

Keywords:
Non-photorealistic rendering (NPR)
Natural media simulation
Pencil rendering
Dynamic optimization yielding voluntary arm movement trajectory
Image processing

ABSTRACT

In applications such as architecture, early design sketches containing accurate line drawings often mislead the target audience. Approximate human-drawn sketches are typically accepted as a better way of demonstrating fundamental design concepts. To this end we have designed an algorithm that creates lines that perceptually resemble human-drawn lines. Our algorithm works directly with input point data and a physically based mathematical model of human arm movement. Our algorithm generates unique lines of arbitrary length given the end points of a line, without relying on a database of human-drawn lines. We found that an observational analysis obtained through various user studies of human lines made a bigger impact on the algorithm than a statistical analysis. Additional studies have shown that the algorithm produces lines that are perceptually indistinguishable from that of a hand-drawn straight pencil line. A further expansion to the system resulted in mimicked dashed lines.

1. Introduction

Non-photorealistically rendered images can convey information more effectively by omitting extraneous detail (abstraction), by focusing the viewer's attention on relevant features (emphasis), and by clarifying, simplifying, and disambiguating shape [14,35]. In fact, a distinguishing feature of non-photorealistic rendering (NPR) is the concept of controlling and displaying detail in an image to enhance communication. The control of image detail is often combined with stylization to evoke the perception of complexity in an image without its explicit representation. NPR imagery, therefore, allows:

- *the communication of uncertainty*: precisely rendered computer graphics imply an exactness and perfection that may overstate the fidelity of a simulation or representation; and
- *the communication of abstract ideas*: simple line drawings, like the force diagrams used in physics textbooks, can communicate abstract ideas in ways that a photograph cannot.

Although there are many current computer-generated drawing techniques that enable the creation of complex stylized images, such stylization techniques are typically limited to a library of previously drawn strokes and may not provide lines with the qualities of expressiveness and aesthetics that match hand-drawn illustrations.

The ultimate goal of this research is to capture the essence of a single stroke, drawn by humans as straight pencil lines of arbitrary length, and encode it into an algorithm. In turn, an application may use this algorithm to produce a line that resembles a human-drawn line, and it could be used to replace traditional computer-drawn lines (e.g., Bresenham's line algorithm [4]).

A challenge in our work is that no precise metric has been developed to generally differentiate between hand-drawn and computer-generated line drawings, although some attempts have been made for specific techniques, such as stippling [21]. However, humans can typically distinguish differences of these results with relative ease [16]. For pencil lines this may be due to changes in grey levels, a variation not proportional to the path, or a glitch in the path orientation. Such variations make it difficult to produce aesthetically pleasing, natural looking results that mimic human-drawn lines.

We divide our algorithm for generating a human-like pencil lines into two parts: (a) synthesizing the path that corresponds to human arm movement and (b) synthesizing the pencil texture applied along the path [15,28,35], inspired by the textures produced by real pencils. Based on this general approach, our algorithm produces high quality simulations of hand-drawn lines,

* Corresponding author at: David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. Tel.: +1 2263392270.
E-mail address: zmeraj@cgl.uwaterloo.ca (Z. AlMeraj).

0097-8493/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2009.04.004

easily incorporated into existing applications, produces lines of arbitrary length and multiple styles, does not require a library of sample lines or user specified attributes (as is the case with [34] technique); it uses only a grey level dynamic range array and a co-occurrence (CC) matrix.

The evaluation of our technique was done by conducting user studies to show that the proposed approach successfully captures and synthesizes aesthetically pleasing lines that mimic hand-drawn lines.

Our contribution, therefore, is a high quality pencil media line reproduction agent for creating aesthetically pleasing lines that mimic human-drawn lines. For this purpose, we use methods of image synthesis and a model of human arm movement for its replication. Our method avoids computationally expensive techniques and large storage space while continuously producing new, unique lines. In addition, the algorithm does not require the setting of user-determined parameters (patterns of deformation, pressure, density, etc.) except for a pencil type selection through the user interface.

A first version of this technique was published in the Computational Aesthetics Conference 2008 [23]. Building on this initial work, more extensive research has been done to analyze and compare real and generated line textures followed by an enhanced explanation of the evaluation strategy. Details involving the original experimental user studies have been added in addition to two algorithms for higher resolution textures and dashed lines.

2. Previous work

Our work draws from research on interactive non-photorealistic rendering methods that approximate artistic hand-drawn images or paintings. We classify this research into three categories: (1) research in human line drawing algorithms that use different style capturing techniques to achieve a replication of a similar stroke containing all of its feature characteristics, (2) texture synthesis methods used in research to reconstruct synthesized images from data taken from natural scenes, and (3) mathematical models replicating unconstrained point-to-point human arm movement trajectories. We review previous work in these three areas in the following.

2.1. Simulating human line drawings

Characteristics of lines in sketched and hand-drawn images have been studied closely in the field of NPR (e.g., [6,16]). Many algorithms captured style characteristics and applied multiple parameters (such as length, width, and pressure). Previous methods [15,28,35] used such style parameters, applied a vector or texture representation of the style to a path, and distorted the piecewise polynomial curve or polygon path to create a stylized line. Other approaches reproduced and synthesized similar styles from example lines [5,9,11,17,19,32].

Our work is inspired by methods that simulate wax crayons (e.g., [26]) and pencils as a medium. Specifically the work by Sousa and Buchanan [33,34] is related to our own, who simulated graphite pencils on a paper medium using a two-level system based on microscopic observations [33]. Their work focused on generating fill strokes, using it for hatching purposes to reproduce artistic drawings. Our method differs because our lines are not restricted to short strokes and can vary greatly in length with no repeating segments. We also base our work on interactive pen-and-ink illustration by Salisbury et al. [27] who described a level-of-detail system that interactively produces multiples of strokes to avoid tediously placing them manually. Our algorithm for drawing lines could easily be incorporated into the above approaches, adding the benefit of a model of human arm movement and a simple perceptual simulation model for graphite pencils without the requirement of a library of lines to copy, paste, and reshape.

Our method differs from the example-based methods explained above in that we do not require example lines to generate unique paths and textures. For each pencil type there exists two pieces of information in the algorithm, a dynamic range, and a co-occurrence matrix. We simply only require vector endpoints to produce lines. Our aim is not to generate varying style from a single given style as seen in example-based methods, but to develop an algorithm that generates lines that vary in path orientation and texture synthesis, mimicking human movement and graphite pencil deposits observed from real straight line drawings on paper.

2.2. Texture synthesis

We were also influenced by a texture synthesis method based upon sequential synthesis [12] that creates a new texture by preserving the second-order statistics of the natural texture into the newly synthesized texture. Gagalowicz and Ma [12] also provided experimental results demonstrating that the visual system is only sensitive to second-order spatial averages of a given texture field, which is one of the reasons we adopted such a methodology. More recent texture synthesis research renames second-order statistics (spatial averages) [12] using the term co-occurrence models or grey level co-occurrence (GLC) models [8]. These are defined as the proportion of second-order probability statistics of pairs of grey levels when their locations differ by a delta in the texture field plane. Copeland et al. [8] used a texture similarity metric based on the texture field of a model texture. The multi-resolution version of their algorithm "spin flip" demonstrated satisfactory performance and resulted in pleasing outputs.

Zalesny and Gool's work [38] also introduced a texture synthesis method based on image intensity statistics by collecting first-order statistics, measured using an intensity histogram, and then extracting the co-occurrence matrix by sampling with point pairs defining a vector. The CCM stores the distribution of intensity differences between the pair of pixels for a given orientation of the vector. The conventional way of calculating the CCM is by summing all the joint probabilities of intensities for a given pixel neighbourhood into their relative position in the CCM. Zalesny and Gool [38] pointed out that the results are better for textures that contain considerable variations, but at some computation cost. In our work we found that the conventional method to acquire the co-occurrence matrix gave adequate results and there was no advantage to be gained using this technique since the textures we use are not so complex.

2.3. Mathematical models of human arm movement

In order to produce a good simulation of a human line drawing, we also examined studies of the coordination of voluntary human arm movements. Human motor production has been analyzed, modelled and documented for well over a century [1,2,29,37]. Over the past few decades, theories of the functions of the central nervous system (CNS) with respect to human arm movement lead to the hypothesis of various mathematical models [3,7,10,20,22,36]. According to these CNS theories, arm movements are produced in either one of two ways: (1) natural movements maintain a constant ratio of angular velocities of joints to bring reduction in control complexity and constrain the degrees of freedom; or (2) hand trajectories are in extra-corporal space, joint

rotations and additional non-physical parameters are tailored to produce the desired or intended hand movements.

Plamondon's model [25] described a synergy of agonist and antagonist neuromuscular systems involved in the production of arm movements. He developed his theory by modeling the impulse responses to neuromuscular activities; his system produced a close proximity bell-shaped velocity profile to represent an entire point-to-point movement.

The *minimum jerk model* introduced by Flash and Hogan [10] formulated an objective function to solve a dynamic optimization problem for measuring the performance of any possible movement as the square of the *jerk* (rate of change of acceleration) of the hand position integrated over an entire movement from start to end positions. Flash and Hogan [10] showed that the unique trajectory of planar, multi-joint arm movements that yields the best performance was in agreement with experimental data. Their analysis was based solely on the kinematics of movement and independent of the dynamics of the musculoskeletal system as in the work done by Plamondon [25].

We adopt the Flash and Hogan model because this model produces trajectories that resemble physical arm movements in a planar field, and because it achieves the best time performance independent of the dynamics of the musculoskeletal system.

3. The human line algorithm (HLA)

Our algorithm combines a number of separate steps or stages in order to successfully synthesize realistic straight and dashed pencil lines. First, we perform a close analysis of lines acquired through user studies of both short, long, and dashed lines. Based on the observational study and a model of human arm movement, we can construct a method to generate a realistic path. We then synthesize a suitable pencil lead texture at a high resolution. Our algorithm has been tested on various forms of input data and verified via a perceptual experiment.

An overview of the complete system is shown in Fig. 1. The first step (Phase A) is a one-time pre-processing stage that captures the statistical properties of hand-drawn lines. Each pencil type is used to draw a model semi-straight (fairly accurate) line that is scanned into the computer; analysis of the line captures grey levels that are stored as a histogram and a co-occurrence matrix.

The second step in the line generation process (Phase B) is as follows: first, the input from the user consists of a pencil type and end points of lines to be drawn. These points can be placed on the screen using mouse or tablet, or read in from text files formatted to include start and end points of all segments (postscript files). The main algorithm consists of the following steps:

(1) Initialize the set of statistical values for the pencil type selected (from the data stored in phase one).
(2) Input user provided point positions into the model of human arm movement to generate line paths.
(3) Synthesize pencil textures and place modified textures along the line to represent a real pencil texture.
(4) Output the final textured lines to the display.

Our approach uses the concept of path and style [35], a human arm trajectory model by Flash and Hogan [10], and a co-occurrence texture synthesis technique similar to that of Gagalowicz and

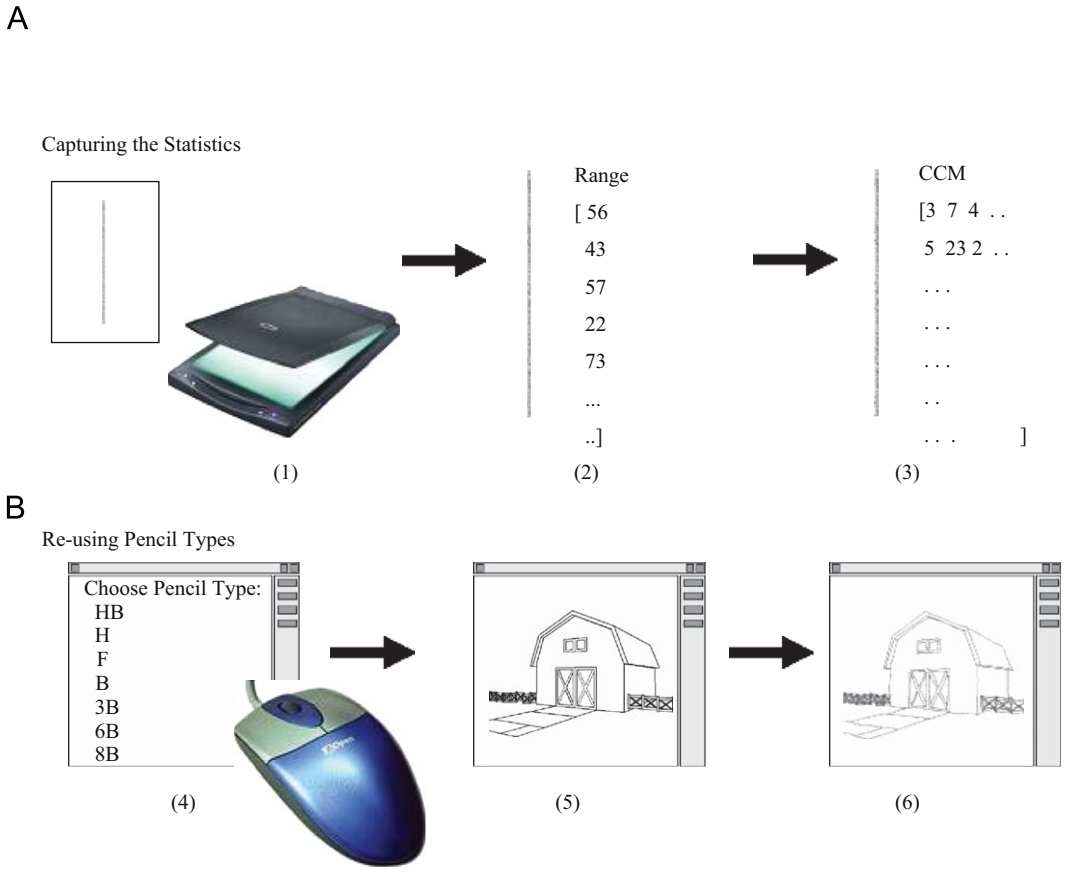

Fig. 1. An overview of the line drawing system. Phase A (top), capturing of the pencil style from an original drawing by scanning it in (1), capturing the dynamic range of the pencil line (2); and (3) capturing the CCM of the line and storing it together with the histogram. The second phase B (bottom), selecting a pencil type in (4); loading a 3D model into 2D space (5); and finally (6) the resulting illustration of applying the selected pencil type to the model.

Ma [12]. The system creates lines that are always unique, both in path and in texture. This is achieved by performing a distribution of texture across the resulting path aligned with its orientation, and using the pencil style co-occurrence matrix to make the line texture closely resemble an original pencil sample. The trajectory path may also require creating a smooth spline along the line depending on the line length.

3.1. Short and long, straight and dashed line observations

This section covers the observations we made when conducting two user studies to observe the hand-drawing of pencil lines. The first of these studies concentrated on solid lines, while the second one analyzed lines with different kinds of dotted and dashed patterns. We compare our observations with results previously presented such as the models by Schlechtweg et al. [28], Sousa and Buchanan [33], or Kalnins et al. [19].

The main aim of both studies was to capture: the pencil texture and path deviations from various path orientations; how the pencil texture appeared for different line orientations; whether the pencil texture properties can be classified into groups depending on line orientation; how the human arm performs its movement when drawing different line orientations; how much the line path is affected by the human arm movement; and the path and texture of dashed lines.

Participants included 12 volunteer students and faculty from the Computer Science Department at the University of Victoria, Canada. A video recorder was used to capture the complete arm movements and drawing process of the participants. This proved very useful when making the decision of whether or not to use a mathematical arm movement model to replicate human movement.

Each study took 3 min to complete on average. Participants were seated in front of a horizontal desk at a reasonable distance (see Fig. 2(a)), and were asked to sit upright, hold the pencil steadily, draw lines at a comfortable pace, not to move the papers position throughout the experiment, not to draw over the lines once already done, not to erase anything, and to adhere to starting and ending points.

The participants were given sheets of paper with circles and squares (Fig. 2(b)). A *circle* defined a starting point and a *square* defined an ending point for a line to be drawn. Each study paper was placed in a frame created on the table to ensure that the user kept the paper in one position as he or she drew. While this also leads to awkward drawing postures at times, we wanted to observe how this affects the line quality, and through our setup we always included comfortable positions as well. The ultimate goal was to collect standard line drawing with the least amount of bias possible.

In the first user study participants drew solid lines that ranged from horizontal, vertical, and angled lines of three different lengths (short (≤ 5 cm), medium (≤ 10 cm), and long (≤ 25 cm)). The objective assigned to participants in this first user study was to draw straight lines as specified between the given dots while not using a dot more than once and to adhere to dot starting and ending points. With the same objectives, the second user study acquired different styles of dashed lines that the user drew horizontally, either from left to right or vice versa. Sample line figures were placed on each sheet of paper to give the participants an exact idea of what was expected.

The ideas collected from this experiment formed the basis of our human line algorithm. For example, when observing straight lines drawn on paper in random orientations from a starting position to an ending position, the amount of deviation found in paths for different line orientations was approximately equal and a large observational significance was not noticed. In addition, we noticed some other irregular and inconsistent characteristics that may be interesting to examine in the future but do not affect the reproduction quality of our lines such as a slight curve near the end of some lines, slight perturbation of the paths to either side of the centre, and variation in pencil darkness along the lines.

The following observations have influenced the implementation of our human line algorithm for pencil line synthesis. We noticed that long lines (≤ 25 cm) tend to have more irregularity along the sides of the pencil texture and deviation from the path. The widths of the lines varied from start to finish (due to pressure and pencil handling), this coincided with the change in grey levels along the line, caused mostly by distributed pressure.

The observations for short lines, medium and dashed lines (≤ 10 cm) revealed that widths of the lines are fairly constant from start to finish; however, the tips of short and dashed lines are less tapered than longer lines. Changes in grey levels along the shorter lines are barely noticeable and the path deviation from the line centre is less evident.

We noticed some commonality between the different dashed lines we collected in our studies, in particular, that the line widths remain fairly constant for each dash. The pencil textures are similar for all dashes, they contain a small percentage of light grey pixels and do not (or very rarely) contain any white pixels. The start and end points of each dash do not always taper. Pencil pressure is slightly higher for alternate dashes. All of the observed paths follow approximately the same path simulated by the Flash and Hogan model for lines.

None of our observations identify distinguishable features for any type of angled line including horizontal and vertical.

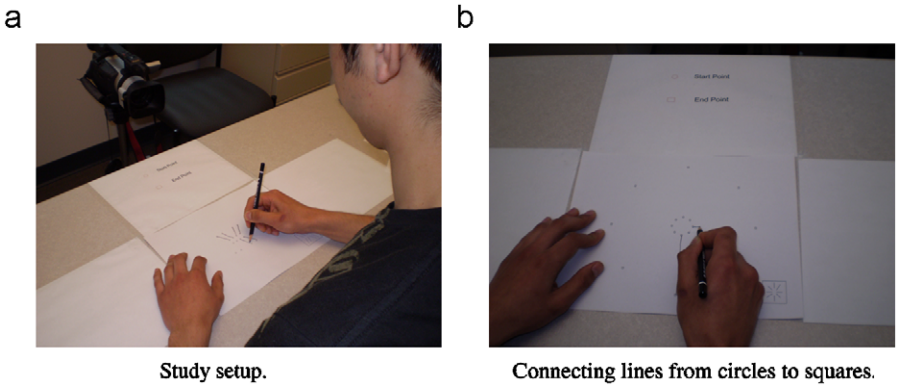

Fig. 2. Pencil drawing study: (a) study setup and (b) connecting lines from circles to squares.

Orientation is also a missing parameter in the Flash and Hogan mathematical model but as our observations show, orientation does not appear to play a role in the ability to create algorithms for reproducing human-like lines.

3.2. Flash and Hogan user studies

Flash and Hogan's model is also based solely on observational conclusions, not on neuro-muscular limitations or facts about how humans move their arms. Flash and Hogan [10] evaluated their model by comparing their simulated trajectories to measured hand trajectories. Planar horizontal arm movements were recorded using an apparatus (see Fig. 3) involving a horizontal table and a mechanical two-link manipulandum held by the hand of the user. The user's shoulders were restrained in all the experiments. The joint angles of the apparatus were measured by precision potentiometers. Data analysis was performed to compute the hand position and joint angles. Lagrange polynomial differentiation was used to calculate joint angular velocities, hand velocities, and curvature of the path.

Flash and Hogan [10] specifically noted the lack of invariance of joint trajectories to translation and rotation of the movements compared to the invariance of hand paths and speed profiles, due to the assumptions they made in the initial optimization stage. Very little discussion was made as to how the lines were actually perceived and how the observational conclusions were drawn. We also noticed the lack of rotational variation in our user studies (Section 3.1) and subsequent evaluation studies (Section 5).

3.3. The path

We construct a path that conforms to a human arm trajectory using the method described by Flash and Hogan [10]. We use this method as it provides "the smoothest motion to bring the hand from an initial position to the final position in a given time" [10]. Our goal is to simulate a hand-drawn line only given two points. The Flash and Hogan model produces trajectories that (1) are invariant under translation and rotation and (2) whose shape does not change with amplitude or duration of the movement. All of these conclusions were based on the observations of low frequency movements of human subjects. No high frequency movements were observed or implemented. Our work follows through with the same assumptions.

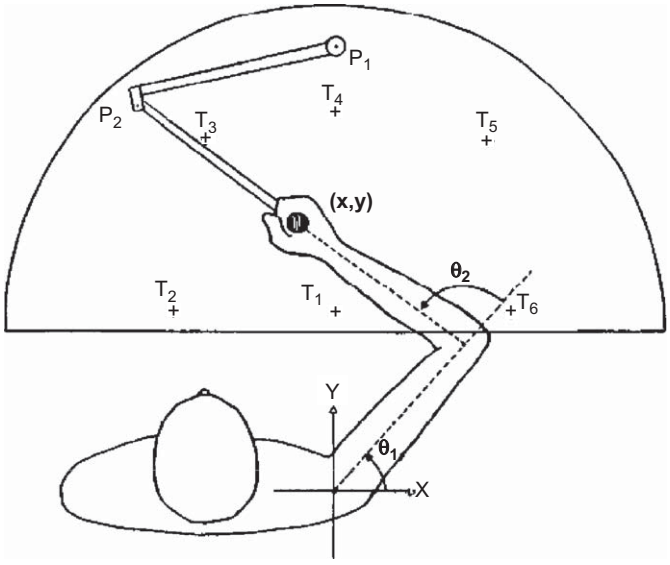

Fig. 3. Apparatus used by Flash and Hogan [10] to measure arm trajectories in a horizontal plane. Movement of the hand was measured using potentiometers P1 and P2.

The mathematical model of Flash and Hogan satisfies our criteria in that it leads to a fast and interactive line path algorithm by providing a realistic overall velocity and acceleration profile, as well as trajectory. Our user studies conducted in Section 3.1 show that the disparity of the line orientation was not noticed by the users and did not affect the overall classification criteria used by participants.

The algorithm for synthesizing a path operates upon two points, representing an initial and a final position of a segment. The points can be entered into the system using one of the three previously mentioned methods.

Every point pair is used to specify the start and end position of the path to be generated by the Flash and Hogan model (Eq. (1)). The mathematical model produces the trajectory (the path) control points that are then used as input for a Catmull–Rom spline.

Our lines are defined by Eq. (1), a fifth-order polynomial:

$$x(t) = x_0 + (x_0 - x_f)(15\tau^4 - 6\tau^5 - 10\tau^3) + D$$
$$y(t) = y_0 + (y_0 - y_f)(15\tau^4 - 6\tau^5 - 10\tau^3) + D \quad (1)$$

where

$$\tau = \frac{t}{t_f}, \quad t \in [0,2] \quad \text{and} \quad D = n, \quad n \in [-5,5]$$

and in which x_0, y_0 are the initial hand position coordinates at time increment $t = 0$, and x_f, y_f are the final hand positions at $t = t_f$. The value of t is the continuous parameter for generating the path, defined in the interval [0,2]. It begins with the start position t_0 and ends with final position t_f. We found empirically that the average time needed to move the hand from an initial position to a final position for the line lengths we specify should be $t_f = 2$. We found empirically that this provides the most satisfactory results.

The trajectory points provide control points for a Catmull–Rom interpolating spline [31]. Catmul–Rom has the advantage of interpolating control points reasonably reproduces the variations of a human line. We chose this over b-splines since b-splines do not interpolate control points and in a high-degree form b-splines are not easy to control. The number of points (defined by the time step, Δt) depends on the length of the line. Experimental evidence [10] shows that short hand-drawn lines are perceptually closer to straight lines and longer lines have more variation. We conducted experiments to find reasonable values for Δt and provide the results in Table 1. Fig. 4 shows the positions of the control points for three lines of varying length using each of the prescribed values for Δt.

The straight line data collected from the first pilot study explained in Section 3.1 showed considerably more variation from the centre line than can be accounted for with the variation of trajectories calculated by Eq. (1).

To represent this variation, we introduce a deviational parameter called *squiggle* (in pixels) as an extension to the Flash and Hogan model.

The squiggle as defined and implemented in previous work by Strothotte and Schlechtweg [35] acts as a control variable; by

Table 1
Empirical values for the time step Δt.

Approx. line length in pixels	Approx. line length in cm	Time step Δt
[0,200]	≤ 5	0.5
(200,400]	≤ 10	0.3
>400	≤ 25	0.2

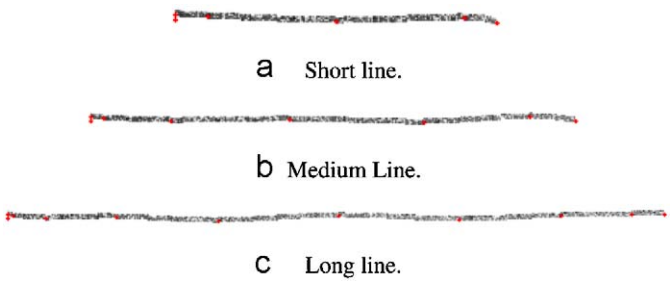

Fig. 4. Generated lines and control point positions for varying lengths. For time steps of (a) $\Delta t = 0.5$, for (b) $\Delta t = 0.3$ and for (c) $\Delta t = 0.2$. The red dots represent the control point.

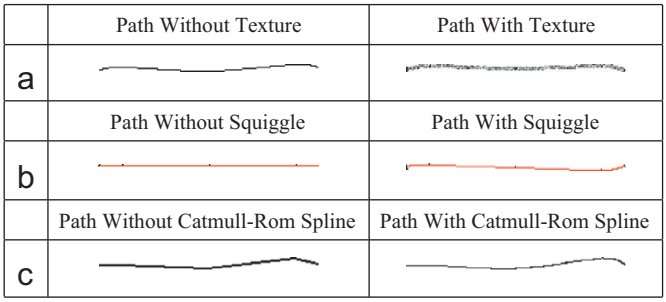

Fig. 5. This figure shows trajectories that can be generated using our path algorithm. It offers a comparison between various trajectories shown (a) with and without texture, (b) with and without squiggle, and (c) with and without a Catmull–Rom interpolating spline incorporated.

setting its value to a maximal amplitude, random displacements are used to give the appearance of a more irregular path. We adopt this as an additional term seeing it necessary to provide sufficient variation in the line paths based on our observations of human-drawn drawings.[1]

We further define *squiggle* as a variable, D, for controlling the deviation approximately normal to the Catmull–Rom spline at each of the lines control points also similar to methods used by Strothotte and Schlechtweg [35]. The magnitude of the *squiggle* parameter controls the magnitude of random displacements applied to the Catmull–Rom spline's controls points and, therefore, strongly influences the appearance of the path.

The deviational value is applied approximately normal to the Catmull–Rom spline at each of its control points. The variable D (see Eq. (1)) varies randomly based on a C++ pseudo-random number generator in order to provide variation along the path, producing independent displacement values for each control point. Empirically we found a range $[-5,5]$ (in pixels) to work for the lines regardless of the line length. The result of applying the deviation factor is shown in Fig. 5, as well as our texturing scheme and spline incorporation.

3.4. The pencil line texture

After having determined the path of the pencil line to draw, we need to generate its texture. Using a texture synthesis technique, our method produces aesthetically pleasing pencil textures that mimic human-drawn lines. This is possible through closely observing the textures of the human-drawn lines gathered in the initial user study as discussed in Section 3.1.

[1] We conducted a number of experiments to validate this choice, see Section 5 for details.

These observations are then incorporated into our own observational model of graphite pencil on paper and used as a basis for our approach. The emphasis is on the graphite texture distribution and natural appeal, not on the paper quality or attributes of graphite deposits. The goal was to capture the natural appearance of graphite texture and to develop a way to semi-accurately display them through a generated process.

As briefly mentioned before, the texture synthesis algorithm is divided into two steps (Fig. 1). Phase A (texture extraction) gathers information from a wide range of graphite pencils—ranging from soft ones to hard ones—from scans of human-drawn lines and creates a statistical profile from these. We then analyze the lines for their statistical information and store them as histograms and co-occurrence matrices. A co-occurrence matrix is defined as the tabulation of how often different combinations of pixel values (intensities) occur in any given image. Based on these data, the pencil type selection, and the user's input lines, Phase B (texture synthesis) generates synthetic pencil textures along the line paths.

Our pencil texture synthesis algorithm starts generating the line texture using a pixel metric (pixels per inch) by distributing random grey values along the line's path, applies the CCM for the pencil type, and finally applies a Gaussian filter. The texturing method used here is inspired by the use of second-order statistics, also called grey level co-occurrence matrices (GLCM) [8,12,38]. Verification later revealed that the synthesized textures represent a striking similarity to human made ones (see Section 5). In the following subsections we provide details about the texture statistic capturing as well as the texture synthesis steps.

3.4.1. Texture extraction

For each pencil type, we extract its statistical profile in the form of a co-occurrence matrix (CCM, a second-order statistics of textures) by analyzing example lines from our study in a pre-processing step. Choosing co-occurrence matrices to synthesize texture has been shown to be extremely efficient in human texture discrimination [18]; the amount of data necessary to achieve the synthesis is very small and the texture can be generated easily to fit all kinds of surface shapes and sizes. Using this method allows us to control the second-order spatial averages of a given texture, since the synthesis is achieved directly from these second-order statistics without the computation of higher order statistics [12]. Also, texture similarity techniques using CCM probability distribution have shown to have high correlation with human perception of textures (the generated images appear to be very similar visually to the original natural textures).

Copeland et al. [8] showed in their work on psychophysical experiments of synthesis algorithms that human vision is sensitive to second-order statistics, where a texture similarity metric was shown to have a high correlation with human perception of textures. Therefore, we used second-order statistics to synthesize lines textures.

In our approach, we scanned texture samples from graphite pencil types 2H, H, HB, F, B, 3B, 6B, 8B and then analyzed them statistically. The pencil textures used for this process were acquired on plain, 20 lb long grain, 100% recycled, A4 smooth white paper.

The scanned texture samples were then analyzed in order to extract their statistical profile. First, we determine the range of grey levels for pixels along the approximate centre of the pencil line (non-repeating Catmull–Rom control points) and record them in a histogram within the range $[min_{mid}, max_{mid}]$. This serves as a starting point to imitate the bell-shaped curve property we noticed when analyzing the statistical information of the sample pencil textures. We discuss how to simulate this effect in the synthetic lines in Section 3.4.2.

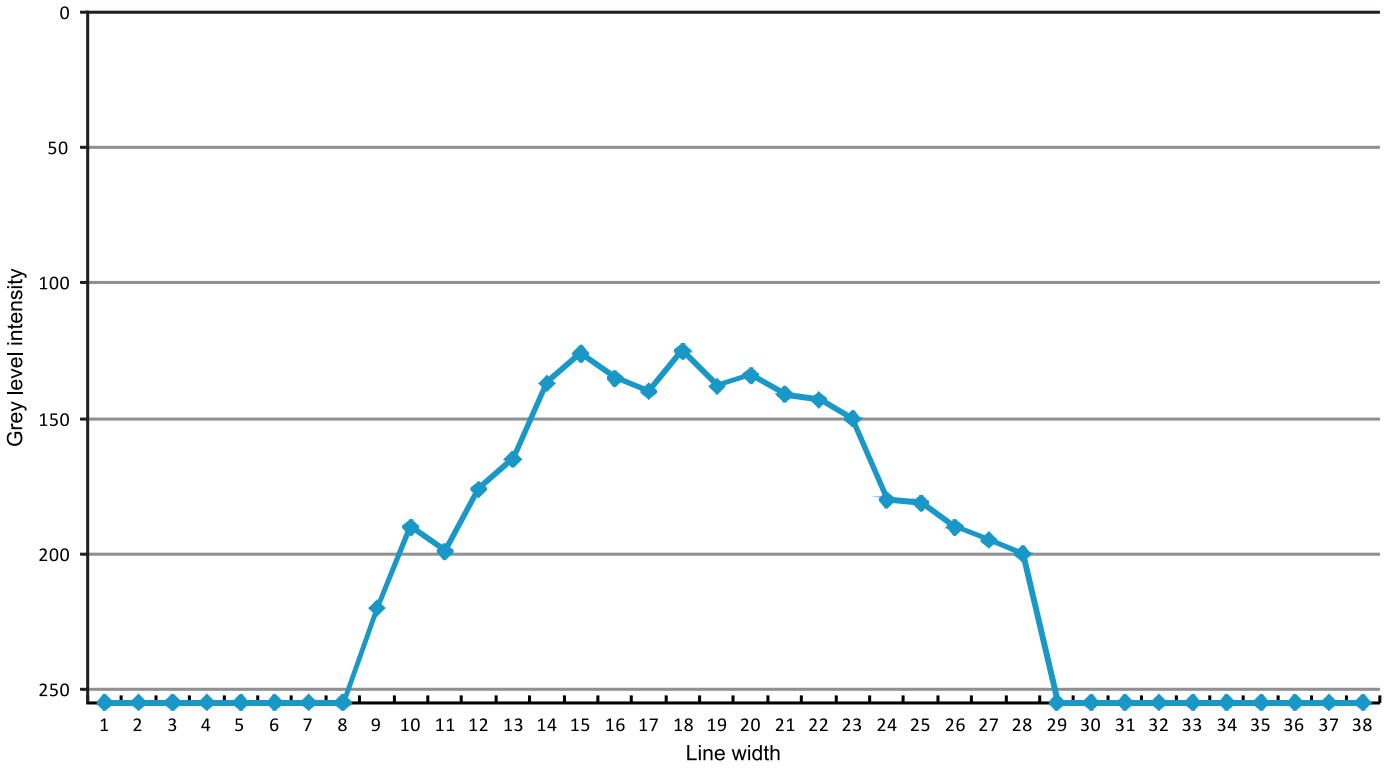

Fig. 6. The approximately bell-shaped curve showing intensity values of cross sections of a line drawn with a 6B pencil.

Fig. 7. Initial synthesized line texture. Placing the grey dynamic range values across the width and the length of the path uniformly.

Next, we determine the histogram of the grey levels for the complete line image, assuming that white pixels surround each line segment when scanned in. The histogram records intensities in the range $[min_{range}, max_{range}]$. This histogram serves as the basis of the lines pencil texture. Finally, we compute the grey level co-occurrence matrix for the line image.

To determine the co-occurrence matrix, each image is acquired on a flatbed scanner so that the line is approximately vertical. The co-occurrence matrix is updated by examining each pixel (p,q) with value $i \in [0,255]$ and its immediate neighbour in the positive y-direction, $(p,q+1)$ with value $j \in [0,255]$. The values (i,j) are indices to the appropriate cell of the co-occurrence matrix C defined over an $n \times m$ image I, parameterized by an offset $(\Delta x, \Delta y) \equiv (0,+1)$ as

$$CCM(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1 & \text{if } I(p,q) = i \\ & \quad \text{and} \quad I(p+\Delta x, q+\Delta y) = j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The CCM probabilities for any single displacement are tabulated in the form of a 256×256 matrix (to ensure that all possible combinations of grey levels are accounted for) with i serving as the row index and j serving as the column index. Since very few white pixels appear inside the line image, we do not consider the white pixels of the histogram when generating the pencil textures.

3.4.2. Texture synthesis

We use both the histogram and the co-occurrence statistics of a target pencil texture to synthesize new textures. We formulate our own grey value distribution technique according to statistical observations that indicate, for most pencil lines, the distribution of grey pixel values starts at a dark value in the middle of the line and falls off according to the bell-shaped curve in the direction normal to the line. This can be seen in Fig. 6. From analyzing scanned lines we used visualization techniques using Matlab software to view the complete data set in 3D view.

First, we fill the line with initial random values (using the histogram for the pre-selected pencil type) and then apply the co-occurrence matrix. The initial line fill includes a random grey value fill of all points along the line and limited to the predefined width.

According to the bell-shaped fall-off that we observed, we distribute grey levels across the centre pixels of the generated spline, replicating the histogram in the range $[min_{mid}, max_{mid}]$. The approximate direction normal to the line is determined and the pixels along this direction to either side of the centre pixel are set to values randomly chosen in the range $[min_{range}, max_{mid}]$ and pixels further to the side of the path are given a lighter intensity in the range $[max_{mid}, max_{range}]$,[2] following the example illustrated in Fig. 7.

The random function defined uses a flat, uniform distribution. This serves its purpose since the second step of the algorithm applies pixel manipulations to create a texture using the CCM.

Next, the co-occurrence matrix is used to re-organize the initial random values to a more suitable pattern that causes the synthesis of a texture that is similar to the sample texture.

[2] Due to the wide range of grey pixel values extracted from real line, these intensity ranges may overlap.

Fig. 8. The values of the pixels are analyzed and pairs of pixels are compared to their indices in the co-occurrence matrix and replaced with an appropriate value if their pair does not exist.

Fig. 9. A 3×3 single-pass Gaussian filter is then applied resulting in the final pencil line texture.

The ultimate goal of this process is to produce a final synthesis with a quality that has more visual similarity to the target texture, and not focus as much on the numerical measure of error between the CCMs of the synthesized and model texture as in Copeland et al. [8]. We achieve this through synthesizing a texture from a CCM.

Next, the CCM is applied by comparing a 3×3 neighbourhood of pixels. If the combination of pixel intensities does not exist in the CCM, the neighbouring pixel's grey value is replaced with an existing neighbouring value from the co-occurrence matrix. The CCM encodes the fact that a neighbourhood relation exists by assigning 1 to $CCM(i,j)$ where i is the grey and index value of the point and j is the grey and pixel value of the neighbour, and assigning 0 to $CCM(i,j)$ otherwise. Because the model textures are taken from lines of set lengths, its is not possible to calculate a quantitative measure of error between the two matrices, the synthesized and the model.

This co-occurrence correction process is repeated over the entire line multiple times until the amount of pixel changes for each step reaches a minimum (pixel combinations equal zero), indicating how well the synthesized co-occurrence matrix represents the model CCM. An example result is shown in Fig. 8.

Once the CCM correction stage is complete, a final step is performed. We apply a single-pass Gaussian filter to the generated lines to reduce tone differences and filter out aliasing effects (Fig. 9). Further investigation of this method could better address line aliasing, such as using a multi-pass Gaussian filter, to enhance the quality of the presented lines. However, even without such further adaptation, our texture synthesis model enables the synthesis of perceptually convincing textures of a pencil line as shown in Fig. 9.[3]

3.5. Pencil resolution

In order to improve the results of our pencil line synthesis technique we examined how the scan resolution affects the results. For this purpose we obtained higher resolution scans of hand-drawn pencil lines and re-analyzed them using our procedure. In addition to the previously described technique explained in Section 3.4.2, we also addressed the small changes in line width that we observed in hand-drawn samples. While the line images in the previous sections have been scanned in at around 100 ppi and generated at the same resolution, the new samples were scanned and images were synthesized at 800 ppi.

After each scan, we slightly changed the scanning procedure and adjusted the brightness levels of the scanned textures. This became necessary because we noticed that scanners automatically change the brightness levels in individual scans. Therefore, to match an approximate target range of grey values we scanned in a grey level ramp at extremely high resolution 2400 ppi, with 16 bit grey scale. Necessary adjustments in PhotoShop were applied to calibrate the scanned grey levels to the approximate brightness of the grey level ramp, this is done by comparing the ramp to each image and adjusting the gamma levels to the same brightness. We then use this second set of adjusted pencil textures as a foundation for our higher resolution line generations.

We used an LCD screen as an output medium. The resulting grey levels of higher resolution images appear better in print than

[3] In Section 4 we show more results of the synthetic pencil line algorithm along with comparisons to real drawn textures.

Pencil	Real	Generated High Resolution
2H		
HB		
B		
6B		

Fig. 10. This figure shows real and generated high resolution pencil textures for a few of the pencil types. The grey levels have been modified and result in better screen and printed image quality. The generated images also include Perlin noise to enhance realism of the generated line texture through randomizing the line width slightly along the path.

Table 2
Empirical values for the gap lengths for the three styles of dashed lines used in this algorithm.

Approx. dash lengths	Dash length Type 1	Dash length Type 2	Dash length Type 3
[0,200]	15–25	15–25	10–20
(200,400]	20–30	25–30	20–30
>400	25–45	35–40	30–45

All units are in pixels.

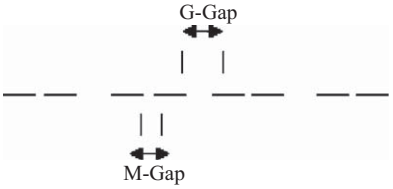

Fig. 11. Dashed line labeling scheme.

lower resolution images, but still more work needs to be done to improve the quality of the line generation for printing. We leave this as a problem for future work.

In addition, we applied line width variations to achieve a more realistic effect of the pencil textures. This was accomplished by applying Perlin noise to the width parameter of the synthesized line textures to obtain a more irregular appearance along the sides of the generated pencil texture. The specific Perlin noise function we use is made up of adding several noise functions together and gives the effect of natural randomness [24], this is then added to the sides of the pencil texture to give a more natural appeal. A selection of final results is shown in Fig. 10.

Table 3
Three types of dashed lines used for synthesis: single, double, and triple dashed lines.

Type	Example Dashed Lines
1	
2	
3	

Table 4
Examples of real and generated dashed lines for the three dashing types.

Real	Generated

Line type	1.Real human line	2."HLA"-generated lines	3."HLA"-generated lines, no CCM
H			
HB			
B			
3B			
6B			
8B			

Fig. 12. Line samples: comparison of hand-drawn lines with synthesized lines and synthesized lines not using the CCM (deviation parameter set to zero).

3.6. Dashed lines

As another aspect of pencil line drawing, our approach can reproduce dotted or dashed lines. People use dashed lines, for example, to differentiate various parts of a drawing. To be able to mimic these styles we used the results of the second part of our first user study that collected various hand-drawn examples of different types of dashed lines (see Section 3.1). Once acquired, the lines were also scanned, brightness re-adjusted, and processed for their statistical information (histogram and CCM) as were the continuous lines. The overall grey level pixel average for the dashes is used to generate a dashed line. In the user study, three types of dashing schemes were acquired from the participants as seen in Table 3. This is an initial attempt to study dashed lines, and further studies could be made to look closer at more styles of dashed or dotted lines.

Closer observations on the different features of the human-drawn dashed lines from our studies resulted in clear distinctions between three dashing schemes: for the dashed lines of Type 1, the spaces between the dashes are approximately as long as the dash itself and are repeated consecutively. The grey levels of the dashes are all applied with similar pressure. The dashes most often miss one of the endpoints. However, the spaces between each consecutive dashes for the Type 2 scheme are slightly larger than the spaces between each two dashes. We observed that the pressure of each consecutive dash is slightly darker than that of the first in the hand-drawn lines. Type 3 dashed lines have relatively short spaces between every three dashes while the space between groups of three dashes is relatively larger (Table 2). In Table 4 we show examples for how our synthesized dashed lines compare to real hand-drawn dashed lines.

We assign dash lengths depending on the length of the dashed lines. Samples of dashed lines that were drawn by our participants showed that dashes in longer lines are drawn with longer strides and larger spaces. Shorter lines are drawn with shorter dashes and shorter spaces in general. Fig. 11 illustrates how we divide and label the dashed lines for replication. We call the smaller gaps between dashes that are found in type two and three of the dashed lines, M-gaps (median gaps). The larger gaps found in all the styles in Table 3 are called G-gaps (group gaps). We conducted

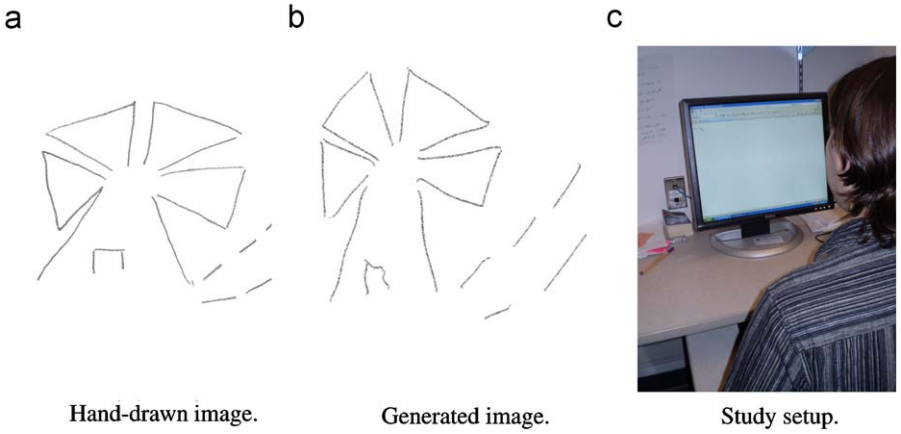

Fig. 13. In our verification study we asked participants whether pencil images were hand-drawn or generated. Here we show samples of these images and the experimental setup: (a) Hand-drawn image. (b) General image. (c) Study setup.

experiments to find reasonable values for these lengths and provided the results in Table 2.

The G-gaps are set to be within a range of [50, 70] pixels. This resulted in pleasing dashed images seen in Table 4.

4. Results

The human line drawing system is implemented through the Qt framework in C++ using OpenGL, and runs on a 2.00 GHz Intel dual core processor workstation without any additional hardware assistance. We can interactively draw lines while the system synthesizes the new path and texture. The system has shown to be successful at assisting users in easily producing a variety of images. All line drawing figures in this paper indicated as "generated by HLA" were drawn using our human line algorithm system.

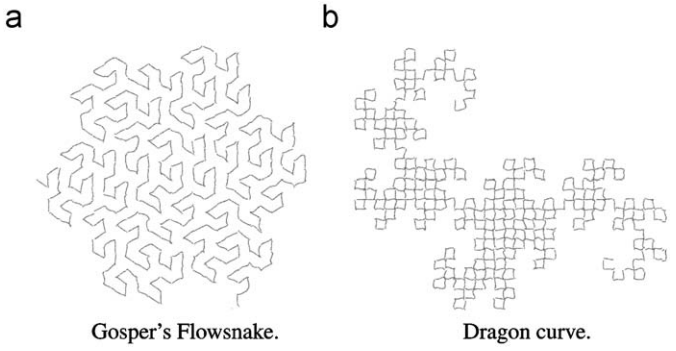

Fig. 14. Two space filling curves using a B pencil generated by HLA. (a) Gosper's Flowsnake and (b) Dragon curve.

Fig. 15. Line hatching generated by HLA (6B pencil).

Currently each pencil line can be rendered using the HLA in 0.24 of a second per line. This is then followed by a postprocess step to apply a Gaussian blur. The speed of the algorithm has lots of room for improvements.

To verify the need for an approach that employs statistics, Fig. 12 shows a comparison between real textures in column 1, generated textures not using a CCM in column 2 (without Step 2 of the synthesis process), and the final synthesized lines in column 3. The placement of all variants side-by-side makes it easier to see the differences between the cases. When analyzed closely, it can be seen that the control of texture composition is more evident in the images of column 3. The CCM applied texture is more homogeneous and big variations between pixel grey values in surrounding neighbourhoods are not noticeable. The textures not using the CCM step in column 2 (only using the grey value distribution technique) are less faithful to textures generated by humans: the white and dark pixel arrangements in these textures give a sense of disorganization and do not represent the smoothness that can be seen in true graphite pencil drawn lines.

5. Verification

We verify the quality of our technique with a study in which participants are asked to decide whether they thought an image was hand-drawn or computer-generated. In this study we concentrate only on the continuous lines to better be able to appreciate the aspect of pencil line texture, rather than the influence of a path or a dashing pattern. Our study uses 18 images, 9 of which were scanned human-made drawings and the other 9 replicate real drawing paths but rendered with our line algorithm. Some samples of the images used and the study setup are shown in Fig. 13. Eleven participants took part in our study, all graduate and undergraduate students in the department. Each participant spent three seconds viewing each image and then selecting true if they thought the drawing was hand-made, false if they thought the drawing was computer-generated. In doing this we also implicitly ask about the line aesthetics, assuming that the hand-drawn aesthetics of sketched lines is our goal, being relevant for a number of applications (e.g., [30]). The viewing time of 3 s was chosen empirically such that participants had enough time to compare drawings without having too much time to over-analyze their decision.

Of the 42% of mistakes made by all subjects, 69.5% of the decisions selected images to be hand-drawn (when they were

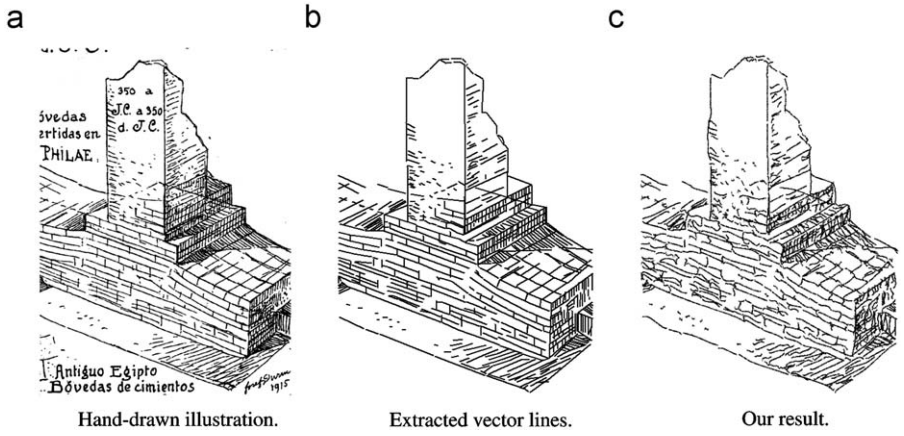

Fig. 16. Our method takes as input lines specified via their start and end points, such as the vector line drawing in (b). The goal is to produce a line drawing that mimics a real hand-drawn graphite pencil image as seen in (a). This is done by modeling human arm trajectory and a statistical model of graphite texture to produce unique, natural looking lines (c), with the aesthetics of hand-drawn pencil lines but without the need for databases of example strokes.

actually computer-generated) and 30.5% of the wrong choices selected images to be computer-generated (when they were actually hand-drawn). This means that the ratio of mistakes that were made of choosing a computer-generated line to be real compared to real lines thought to be computer-generated was about 2:1. A paired sample t-test showed that our computer-generated lines were significantly more often thought to be hand-drawn than hand-drawn lines were thought to be computer-generated (paired $t(10) = 2.849, p < .05$). This result suggest that the computer-generated pencil line drawings were good enough to often pass for hand-drawn ones.

6. Applications

We applied our technique to a number of different example domains. Our line generation (both straight and dashed) can easily be integrated into existing graphics programs that produce line end points, including systems in which the line paths are editable by artists or additional algorithms. Our technique only requires the selection of a pencil type and the specification of the end points for each line. For example, Fig. 14 shows two space-filling curves, Gosper's Flowsnake [13] and the dragon curve, rendered with human-like lines using the B pencil setting.

The synthesized lines are applied to each of the short individual line segments of each curve. The drawings could be improved by detecting co-linear segments and processing them as a single, longer line to better emulate a human drawing.

In Fig. 15, hatching lines are generated and replaced with synthesized lines. We can simulate the effect of hatch-like fill strokes, where each line is unique. The individual unit squares seen in this image were generated using the life game algorithm and filled with a basic hatching method.

In Fig. 16 vector lines are extracted from a hand-drawn illustration and then used as input to the HLA algorithm resulting in the image shown in Fig. 16c. In comparison, the artist who drew the original image (Fig. 16a) seems to have used the assistance of some sort of ruler, while we chose to add more randomness to the lines. This is not a perfect example of the algorithm at work, but is used to show some different effect from the one achieved by HLA.

For enhanced effects, higher resolutions can be obtained by selecting an option in the application interface. Fig. 17a shows an example where the output from a CAD application (an object with 36-edges) was replaced by our lines while Fig. 17b shows the same object rendered with dashed lines. In both cases a simulated 8B pencil was used. Fig. 18 presents an additional example of dashed lines applied to input vectors lines.

Fig. 19 demonstrates an architectural drawing rendered using our lines. This example shows the variability of the line paths when compared with the original straight line drawing, provides a convincing hand-made image.

Finally, by capturing lines (e.g., through tablets) or generating lines from images, the pencil style can be applied to those drawings as well. An example of the latter scenario is shown in Fig. 20 in which the lines extracted from a pixel image were rendered using our techniques.

In summary, we have applied our work in this paper to four genres: space-filling curves, architectural drawings (e.g., CAD, Google's sketch up), line patterns generated from the life game, and reproductions of artist's drawings.

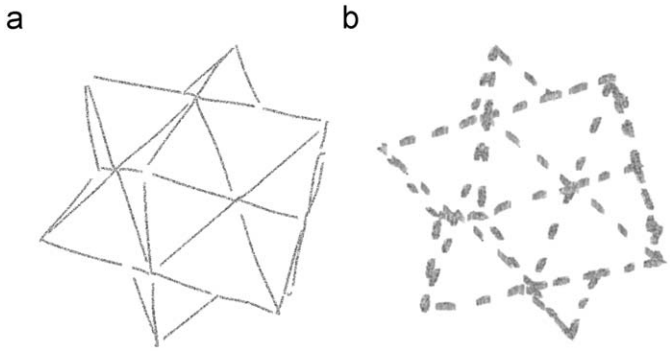

Fig. 17. A 36-sided object generated by HLA with a simulated 8B pencil, both solid (low resolution) and dashed (higher resolution).

7. Conclusion and future work

The main contribution of this work is a technique and system that can synthesize high quality and aesthetically pleasing pencil lines, mimicking those drawn by humans. Our technique is based on the observations of human-drawing pencil lines, in terms of both the line trajectories observed and the textures this process produces. Our algorithm combines texture synthesis with a model of human arm movement to generate unique textures and paths for each line processed. The algorithm avoids computationally expensive techniques and large storage space, thus can easily be applied interactively as an additional step to any process that produces continuously changing unique lines. We verified the results our system produces with a user study which suggested that our computer-generated pencil line drawings were good enough to often pass for hand-drawn line drawings.

More investigation is needed to mimic the appearance of pressure along the lines. Choosing parameters that will make the lines appear to have more character will also increase the aesthetic property of the lines. While we have temporarily included a squiggle factor to compensate for the extra deviation found in longer lines, we hope that through further research we can determine how the human field of view or other factors contribute to the waviness problem of longer lines. More studies should be conducted to rectify problems with print quality.

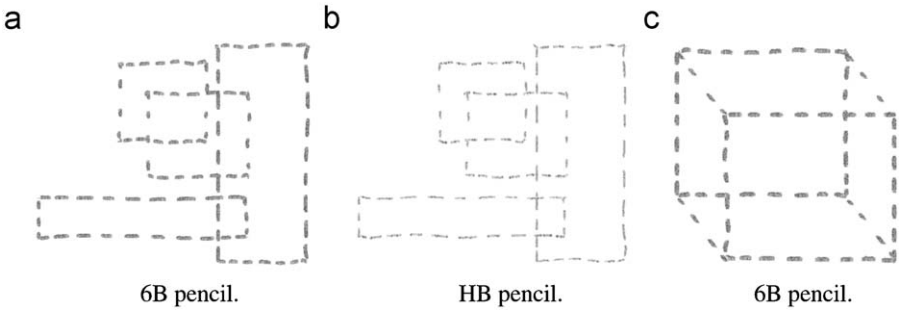

Fig. 18. Dashed line representation of a group of rectangles and of a cube in higher resolution: (a) 6B Pencil. (b) HB Pencil. (c) 6B Pencil.

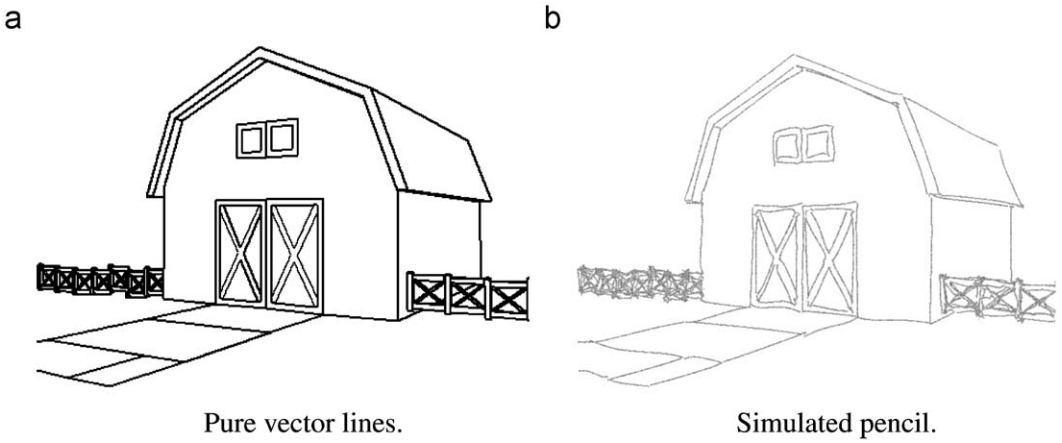

Fig. 19. A barn generated by HLA (H pencil in low resolution): (a) pure vector lines and (b) simulated pencil.

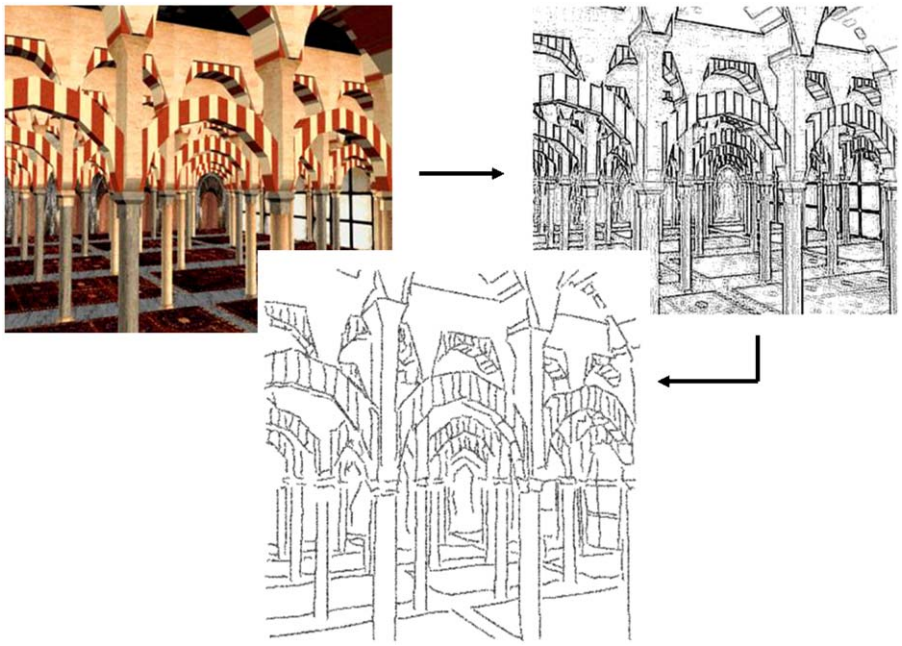

Fig. 20. Using lines extracted from a photograph with HLA: first, the image is filtered to extract a version in which edges are emphasized; then, the result is vectorized, finally, line output of this step is rendered using a simulated H type pencil.

Fig. 21. Example curve generated by HLA, using the Flash and Hogan curve optimization methods, inspired by Finkelstein and Salesin [9].

Similar approaches to the work documented here may work for drawing curved lines (an initial result is shown in Fig. 21), but further investigation would be necessary to correctly mimic the resulting graphite textures on curved paths. Also more work can be done to mimic human hatching through similar observation-based studies, recording the relationship between arm and wrist movement when producing shorter strokes. Fig. 15 is our first attempt to consider hatching as an application; more experiments and analysis are needed to present accurate human hatching techniques.

Acknowledgements

We would like to thank all who participated in our study and to the many who helped make this work complete. This work was partly supported by the National Science and Engineering Research Council of Canada, the University of Victoria, and Kuwait University.

References

[1] Adams J. A closed-loop theory of motor behavior. Journal of Motor Behavior 1971;3:111–49.

[2] Bernstein N. The co-ordination and regulation of movements. London: Pergamon Press; 1967.

[3] Bizzi E, Mussa-Ivaldi F, Giszter S. Computations underlying the execution of movement: a biological perspective. Science 1991;253(5017):287–91 ⟨http://dx.doi.org/10.1126/science.1857964⟩.

[4] Bresenham J. Algorithm for computer control of a digital plotter. IBM Systems Journal 1965;4(1):25–30 ⟨http://doi.acm.org/10.1145/280811.280913⟩.

[5] Brunn M. Curve synthesis by example. Master's thesis. University of Calgary, 2006.

[6] Cole F, Golovinskiy A, Limpaecher A, Barros HS, Finkelstein A, Funkhouser T, et al. Where do people draw lines? ACM Transactions on Graphics 2008;27(3). Article No. 88, ⟨http://doi.acm.org/10.1145/1399504.1360687⟩.

[7] Contreras-Vidal J, Grossberg S, Bullock D. A neural model of cerebellar learning for arm movement control: cortico-spino-cerebellar dynamics. Learning and Memory 1997;3:475–502 ⟨http://dx.doi.org/10.1101/lm.3.6.475.

[8] Copeland AC, Ravichandran G, Trivedi MM. Texture synthesis using gray-level co-occurrence models: algorithms, experimental analysis and psychophysical support. Optical Engineering 2001;40(11):2655–73 ⟨http://dx.doi.org/10.1117/1.1412851⟩.

[9] Finkelstein A, Salesin DH. Multiresolution curves. In: Proceedings of SIGGRAPH. New York: ACM Press; 1994. p. 261–8 ⟨http://doi.acm.org/10.1145/192161.192223⟩.

[10] Flash T, Hogan N. The coordination of arm movements: an experimentally confirmed mathematical model. The Journal of Neuroscience 1985;5(7):1688–703.

[11] Freeman WT, Tenenbaum JB, Pasztor EC. Learning style translation for the lines of a drawing. ACM Transactions on Graphics 2003;22(1):33–46 ⟨http://doi.acm.org/10.1145/588272.588277⟩.

[12] Gagalowicz A, Ma SD. Sequential synthesis of natural textures. Computer Vision, Graphics, and Image Processing 1985;30(3):289–315 ⟨http://dx.doi.org/10.1016/0734-189X(85)90162-8⟩.

[13] Gardner. Mathematical games: in which ''monster'' curves force redefinition of the word ''curve''. SCIAM: Scientific American 1976;235:124–33.

[14] Gooch B, Gooch AA. Non-photorealistic rendering. A K Peters, Ltd., Natick, 2001. ⟨http://doi.acm.org/10.1145/558817⟩.

[15] Hsu SC, Lee IHH. Drawing and animation using skeletal strokes. In: Proceedings of SIGGRAPH. New York: ACM Press; 1994. p. 109–18. ⟨http://doi.acm.org/10.1145/192161.192186⟩.

[16] Isenberg T, Neumann P, Carpendale S, Sousa MC, Jorge JA. Non-photorealistic rendering in context: an observational study. In: Proceedings of NPAR. New York: ACM Press; 2006. p. 115–26 ⟨http://doi.acm.org/10.1145/1124728.1124747⟩.

[17] Jodoin P-M, Epstein E, Granger-Piché M, Ostromoukhov V. Hatching by example: a statistical approach. In: Proceedings of NPAR. New York: ACM Press; 2002. p. 29–36 ⟨http://doi.acm.org/10.1145/508530.508536⟩.

[18] Julesz B, Bergen R. Textons, the fundamental elements in preattentive vision and perception of textures. In: RCV87, 1987. p. 243–56.

[19] Kalnins RD, Markosian L, Meier BJ, Kowalski MA, Lee JC, Davidson PL, et al. WYSIWYG NPR: drawing strokes directly on 3D models. ACM Transactions on Graphics 2002;21(3):755–62 ⟨http://doi.acm.org/10.1145/566654.566648⟩.

[20] Kawato M, Gomi H. The cerebellum and VOR/OKR learning models. Trends in Neurosciences 1992;15(11):445–53 ⟨http://dx.doi.org/10.1016/0166-2236(92)90008-V⟩.

[21] Maciejewski R, Isenberg T, Andrews WM, Ebert DS, Sousa MC, Chen W. Measuring stipple aesthetics in hand-drawn and computer-generated images. Computer Graphics & Applications 2008;28(2):62–74 ⟨http://doi.ieeecomputersociety.org/10.1109/MCG.2008.35⟩.

[22] Mazzoni P, Andersen R, Jordan M. A more biologically plausible learning rule for neural networks. Proceedings of the National Academy of Sciences 1991;88(10):4433–7 ⟨http://dx.doi.org/10.1073/pnas.88.10.4433⟩.

[23] Meraj Z, Wyvill B, Isenberg T, Gooch A, Guy R. Mimicking hand-drawn pencil lines. In: CAe '08, Proceedings of the international symposium on computational aesthetics in graphics, visualization, and imaging (CAe 2008, June 18–20, 2008, Lisbon, Portugal), Aire-la-Ville, Switzerland, 2008. p. 73–80. Eurographics Association. ⟨http://dx.doi.org/10.2312/COMPAESTH/COMPAESTH08/073-080⟩.

[24] Perlin K. Improving noise. In: Proceedings of SIGGRAPH. New York. ACM Press; 2002. p. 681–2 ⟨http://dx.doi.org/10.1145/566570.566636⟩.

[25] Plamondon R. A kinematic theory of rapid human movements, parts I and II. Biological Cybernetics 1995;72(4):295–307 p. 307–20 ⟨http://dx.doi.org/10.1007/BF00202785⟩.

[26] Rudolf D, Mould D, Neufeld E. Simulating wax crayons. Proceedings of Pacific Graphics 2003. Los Alamitos, CA: IEEE Computer Society, IEEE; 2003. p. 163–75 ⟨http://dx.doi.org/10.1109/PCCGA.2003.1238258⟩.

[27] Salisbury MP, Anderson SE, Barzel R, Salesin DH. Interactive pen-and-ink illustration. In: Proceedings of SIGGRAPH. New York: ACM Press; 1994. p. 101–8 ⟨http://doi.acm.org/10.1145/192161.192185⟩.

[28] Schlechtweg S, Schönwälder B, Schumann L, Strothotte T. Surfaces to lines: rendering rich line drawings. In: Proceedings of WSCG, vol. 2, 1998. p. 354–61. URL ⟨http://wscg.zcu.cz/wscg98/papers98/schlechtwegwscg98.ps.gz⟩.

[29] Schmidt RA. A schema theory of discrete motor skill learning. Psychological Review 1975;82(4).

[30] Schumann J, Strothotte T, Raab A, Laser S. Assessing the effect of non-photorealistic rendered images in CAD. In: Proceedings of CHI. New York: ACM Press; 1996. p. 35–42 ⟨http://doi.acm.org/10.1145/238386.238398⟩.

[31] Shirley P, Ashikhmin M, Gleicher M, Marschner S, Reinhard E, Sung K, et al. Fundamentals of computer graphics, 2nd ed. Natick, MA: A. K. Peters, Ltd.; 2005.

[32] Simhon S, Dudek G. Sketch interpretation and refinement using statistical models. Rendering techniques, Aire-la-Ville, Switzerland, 2004. p. 23–32. EUROGRAPHICS association. URL ⟨http://www.eg.org/EG/DL/WS/EGWR/EGSR04/023-032.pdf.abstract.pdf;internal&action=paperabstract.action⟩.

[33] Sousa MC, Buchanan JW. Computer-generated graphite pencil rendering of 3D polygonal models. Computer Graphics Forum 1999;18(3):195–207 ⟨http://dx.doi.org/10.1111/1467-8659.00340⟩.

[34] Sousa MC, Buchanan JW. Observational model of graphite pencil materials. Computer Graphics Forum 2000;19(1):27–49 ⟨http://dx.doi.org/10.1111/1467-8659.00386⟩.

[35] Strothotte T, Schlechtweg S. Non-photorealistic computer graphics: modeling, animation, and rendering. San Francisco: Morgan Kaufmann Publishers; 2002 ⟨http://doi.acm.org/10.1145/544522⟩.

[36] Uno Y, Kawato M, Suzuki R. Formation and control of optimal trajectory in human multijoint arm movement. Biological Cybernetics 1989;61(2):89–101 ⟨http://dx.doi.org/10.1007/BF00204593⟩.

[37] Woodworth R. The accuracy of voluntary movement. Psychological Review: Monograph Supplements 1899;3:1–114.

[38] Zalesny A, Gool LV. A compact model for viewpoint dependant texture synthesis. In: Proceedings of SMILE, LNCS, vol. 2018. Berlin: Springer; 2001. p. 124–43. ⟨http:dx.doi.org/10.1007/3-540-45296-6⟩.